Grafika Komputerowa i Komunikacja Człowiek-Komputer

Data zajęć: 26.11.2018 Data oddania: 15.12.2018
Termin zajęć: Pon. TP 9:15 Prowadzący zajęcia: Mgr inż. Szymon Datko

Sprawozdanie nr 2

Jakub Majewski 238902

---

1. Opis tematu

Temat: OpenGL - modelowanie obiektów 3-D.

Zrealizowane zadania:
1. Zapoznanie się z podstawowymi funkcjami OpenGL odpowiadającymi za renderowanie obiektów 3D.
2. Napisanie programu renderującego obiekt 3D (jajko) w postaci powierzchni opisanej równaniami parametrycznymi.
   a. Renderowanie chmury punktów tworzących kształt jajka.
   b. Połączenie powstałych punktów liniami w celu utworzenia siatki.
   c. Wypełnienie dziur w siatce w celu utworzenia jednolitego obiektu.

2. Opis najważniejszych fragmentów kodu

1. Metody zwracające wartości funkcji parametrycznych opisujących powierzchnię jajka:

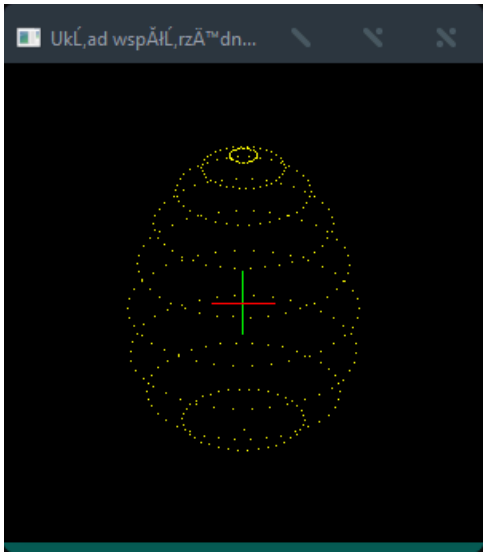
```
1  float Egg::FuncX(float u, float v) {
2    float t[5] = { -90.0, 225.0, -270.0, 180.0, -45.0 }; //
     ↪  Współczynniki wielomianu
3    float f = 0.0; // Tymczasowa wartość funkcji
4    float u2 = u; // Kolejne potęgi argumentu u
5    for (int i = 4; i >= 0; --i) {
6      f += t[i] * u2;
7      u2 *= u;
8    }
9    return f * std::cos(3.14159 * v);
10 }
11
12 float Egg::FuncY(float u, float v) {
13   float t[4] = { 160.0, -320.0, 160.0, 0.0 }, f = 0.0, u2 = u;
14   for (int i = 3; i >= 0; --i) {
15     f += t[i] * u2;
16     u2 *= u;
17   }
18   return f;
19 }
20
21 float Egg::FuncZ(float u, float v) {
22   float t[5] = { -90.0, 225.0, -270.0, 180.0, -45.0 }, f = 0.0,
     ↪  u2 = u;
23   for (int i = 4; i >= 0; --i) {
24     f += t[i] * u2;
25     u2 *= u;
26   }
27   return f * std::sin(3.14159 * v);
28 }
```

2. Metody renderujące obiekt na trzy różne sposoby.

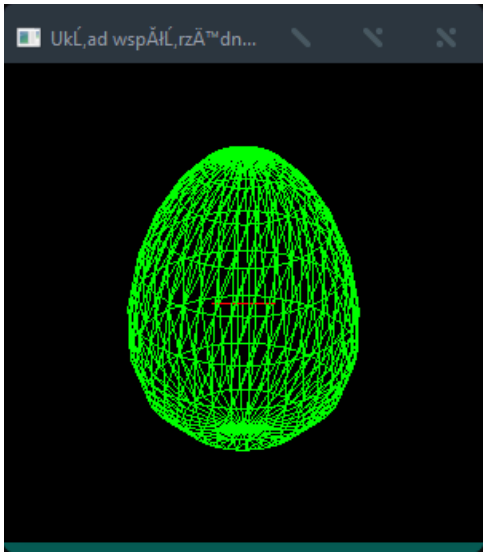
```
// Renderowanie chmary punktów
void Egg::RenderModel_Points() {
  glBegin(GL_POINTS);
  glColor3f(1.0, 1.0, 0.0);
  for (int i = 0; i < N; ++i) {
    for (int j = 0; j < N; ++j) {
      glVertex3d(tab[i][j][0], tab[i][j][1], tab[i][j][2]);
    }
  }
}

// Renderowanie siatki
void Egg::RenderModel_Mesh() {
  glBegin(GL_LINES);
  glColor3f(0.0, 1.0, 0.0);
  for (int i = 0; i < N; ++i) {
    for (int j = 0; j < N-1; ++j) {

      int x = i, y = j;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);
      x = (i + 1) % N;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);

      x = i; y = j;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);
      y = (j + 1) % N;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);

      x = i; y = j;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);
      x = (i + 1) % N; y = (j + 1) % N;
      glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);

    }
  }
}
```

```
// Renderowanie wypełnionej siatki
void Egg::RenderModel_Solid() {
  glBegin(GL_TRIANGLES);
  glColor3f(0.0, 0.0, 0.0);
  for (int i = 0; i < N; ++i) {
    for (int j = 0; j < N-1; ++j) {
      int x = i, y = j;
      putColorPoint(x, y);
      x = (i + 1) % N;
      putColorPoint(x, y);
      y = (j + 1) % N;
      putColorPoint(x, y);

      x = i; y = j;
      putColorPoint(x, y);
      y = (j + 1) % N;
      putColorPoint(x, y);
      x = (i + 1) % N;
      putColorPoint(x, y);
    }
  }
}

// Postawienie punktu na pozycji (x,y) o kolorze zależnym od
↪  położenia.
void Egg::putColorPoint(int x, int y) {
  glColor3f(tab[x][y][0] * 20.0, tab[x][y][1] * 20.0,
  ↪  tab[x][y][2] * 20.0);
  glVertex3d(tab[x][y][0], tab[x][y][1], tab[x][y][2]);
}
```

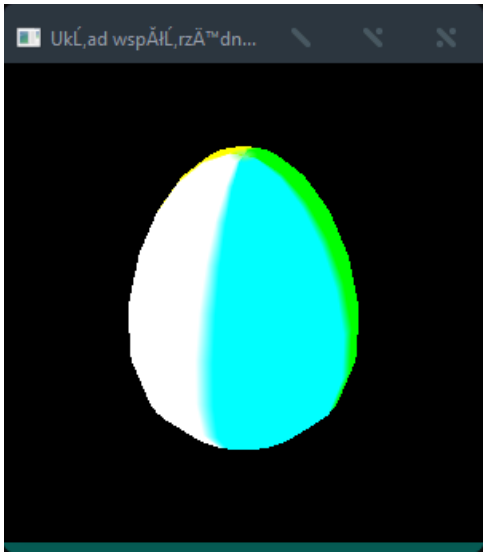
3. Rezultat pracy



Rysunek 1: Efekt wywołania metody Egg::RenderModel_Points()



Rysunek 2: Efekt wywołania metody Egg::RenderModel_Mesh()



Rysunek 3: Efekt wywołania metody Egg::RenderModel_Solid()

4. Spostrzeżenia i wnioski

Podczas testowania i eksperymentowania z kodem okazało się, że duża liczba punktów tworzących płaszczyznę, na której został opisany kształt nie ma znaczącego wpływu na efekt końcowy po nałożeniu siatki i wypełnieniu jej. Największa różnica jest zauważalna przede wszystkim przy renderowaniu samej siatki punktów.
Po nałożeniu kolorów na obiekt zależnych od położenia punktów jajko miało na sobie tylko 4 barwy. Prowadzi to do wniosku, że wszystkie punkty na płaszczyźnie tworzące jajko znajdują się nad płaszczyzną Y.